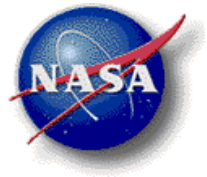


On-Board Autonomy for Rovers

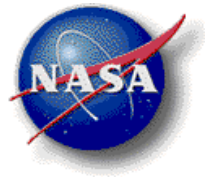
Rich Washington

RIACS / NASA Ames

John Bresina
Howard Cannon

NASA Ames

Outline



- Conditional, flexible execution
  - CRL language and executive
  - Current directions
- Limited plan adaptation
  - Skipping plan steps
  - Plan library for “floating contingencies”

Contingent Rover Language (CRL) overview

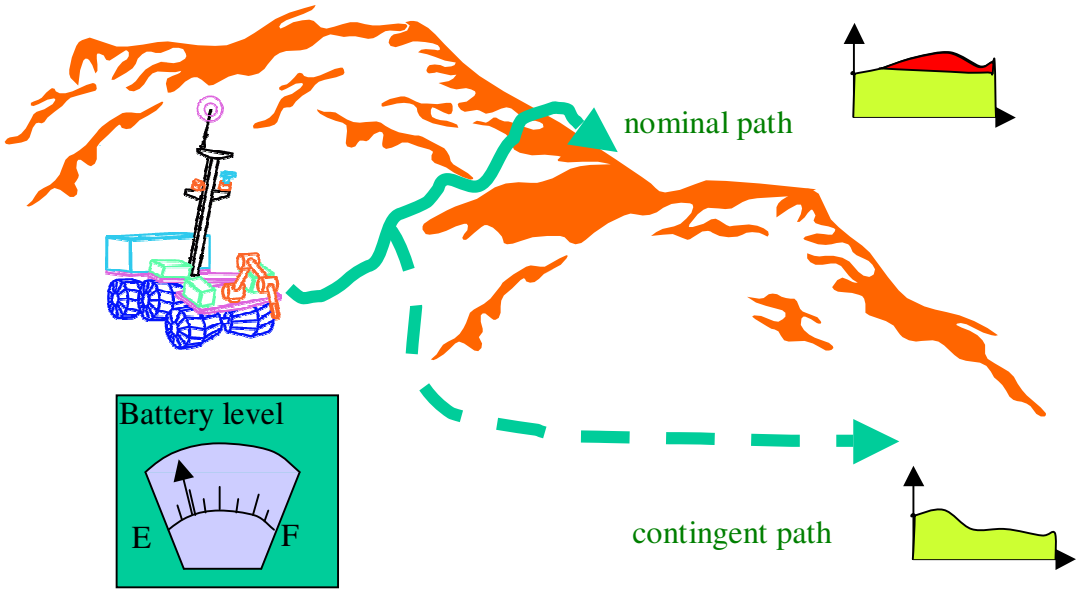
- Flexible, declarative execution language
  - Flexible control structures in a declarative, planner-compatible language
  - Domain-independent, reusable executive

Backward compatibility & new features
  - Allow time-stamped sequences (like Sojourner)
  - Allow relative ordered sequences
  - Additions: branches, flexible time, state and resource conditions
- Utility-based reasoning
- Deployment
  - Feb 1999 field test, Marsokhod: Lisp CRL Exec
  - May 2000 field test, K9: Preliminary C++ CRL Exec
  - Multi-platform
    - Marsokhod, K9, MSF, ATRV, CMU Personal Rover, UAV?

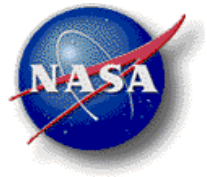
Contingent Rover Language (CRL) overview

- Flexible, condition-based execution
  - temporal conditions (absolute, relative)
  - resource conditions
  - state-based conditions

- Hierarchical structure
  - *task*: executable action
  - *block*: sequence of nodes
  - *branch*: choice point
  - *(concurrent blocks)*



Sample CRL action



Action: (drive ?targetX ?targetY 0.05)

Start temporal conditions: (time 10 300) ← absolute time
(time +5 +20) ← relative w.r.t. previous action

Wait-for conditions: (resource energy 5) ← resource condition

Start conditions: (rover-state :mechanical-state :ok) ← db conditions
(rover-target ?targetX ?targetY)

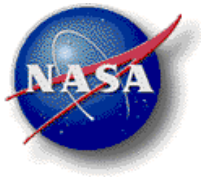
Maintain conditions: (resource energy 2) ← variable

End temporal conditions: (time +0 +600)

Expectations: (duration 100 10) ← mean 100, s.d. 10
(energy 2)

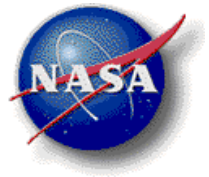
Continue-on-failure: False

Current directions – CRL / flexible execution



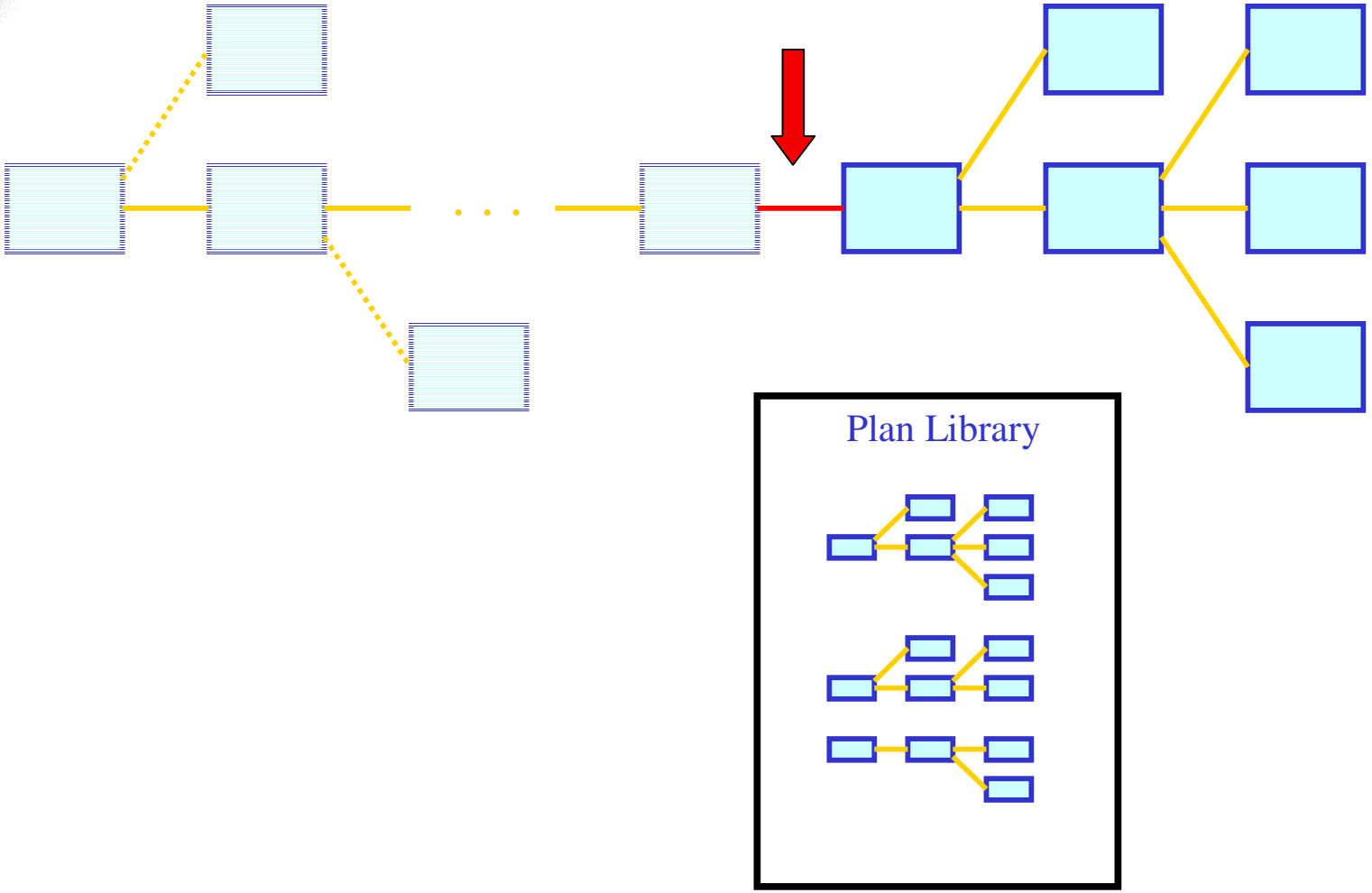
- Limited plan adaptation
  - Floating contingencies
  - Plan step skipping
- Compatibility with planning efforts
  - Concurrent Contingency Planning (Smith, NASA Ames)
    - Concurrent activities
    - Generalization of cross-action temporal constraints
  - Decision-Theoretic Planning for Rovers (Zilberstein, U Mass)
    - MDP-based methods
- Mission infusion

Floating contingencies

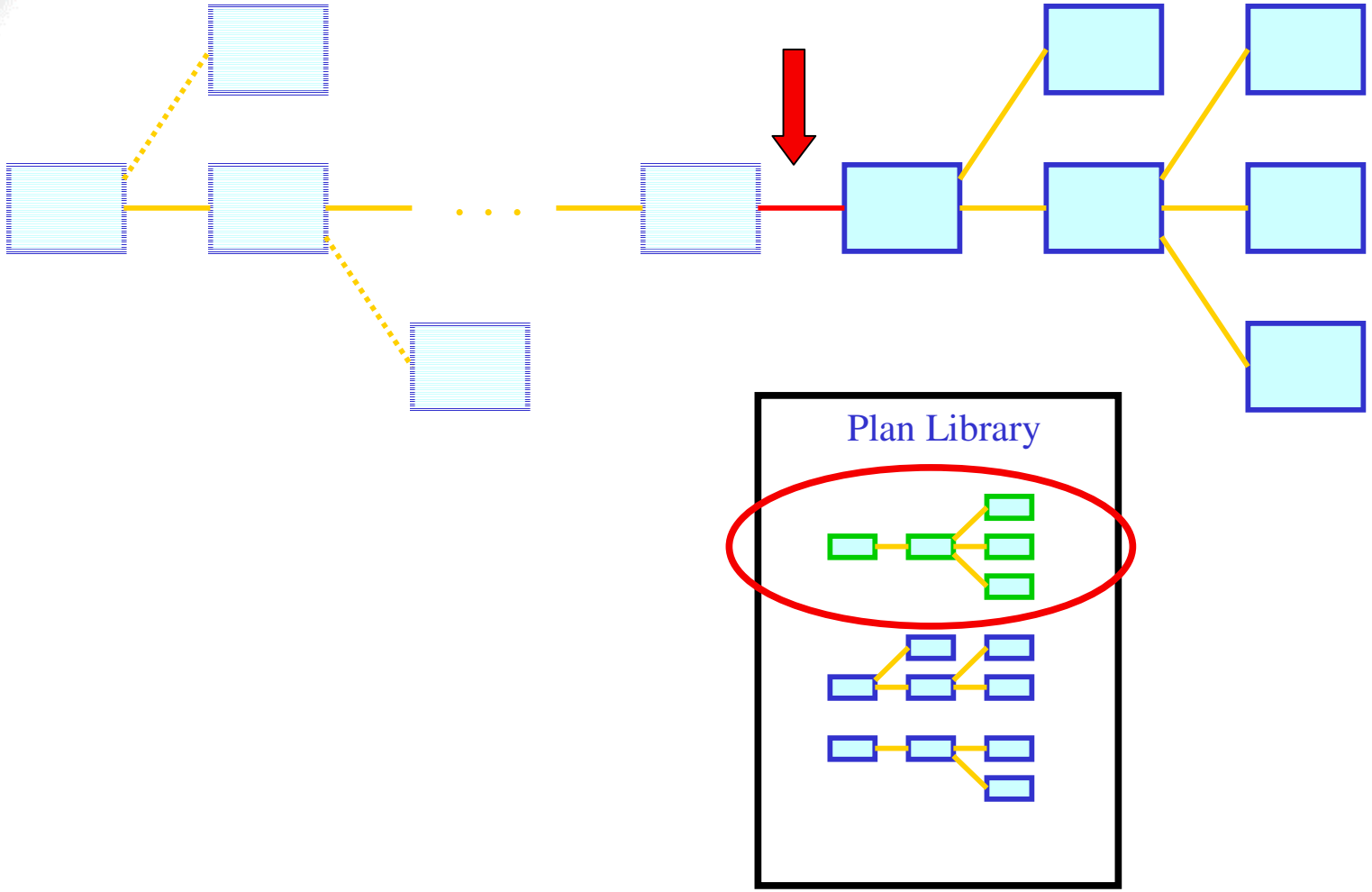


- Library of pre-compiled plans
  - Contingencies
    - Backup plans (call home, perform diagnostics, retry)
    - Alternative methods
  - Opportunities
    - "Unexpected" results of on-board science analysis
- Types of floating contingency plans
  - insert, replace
  - node transition, node failure, continuous
- Choice of contingency plan based on expected utility

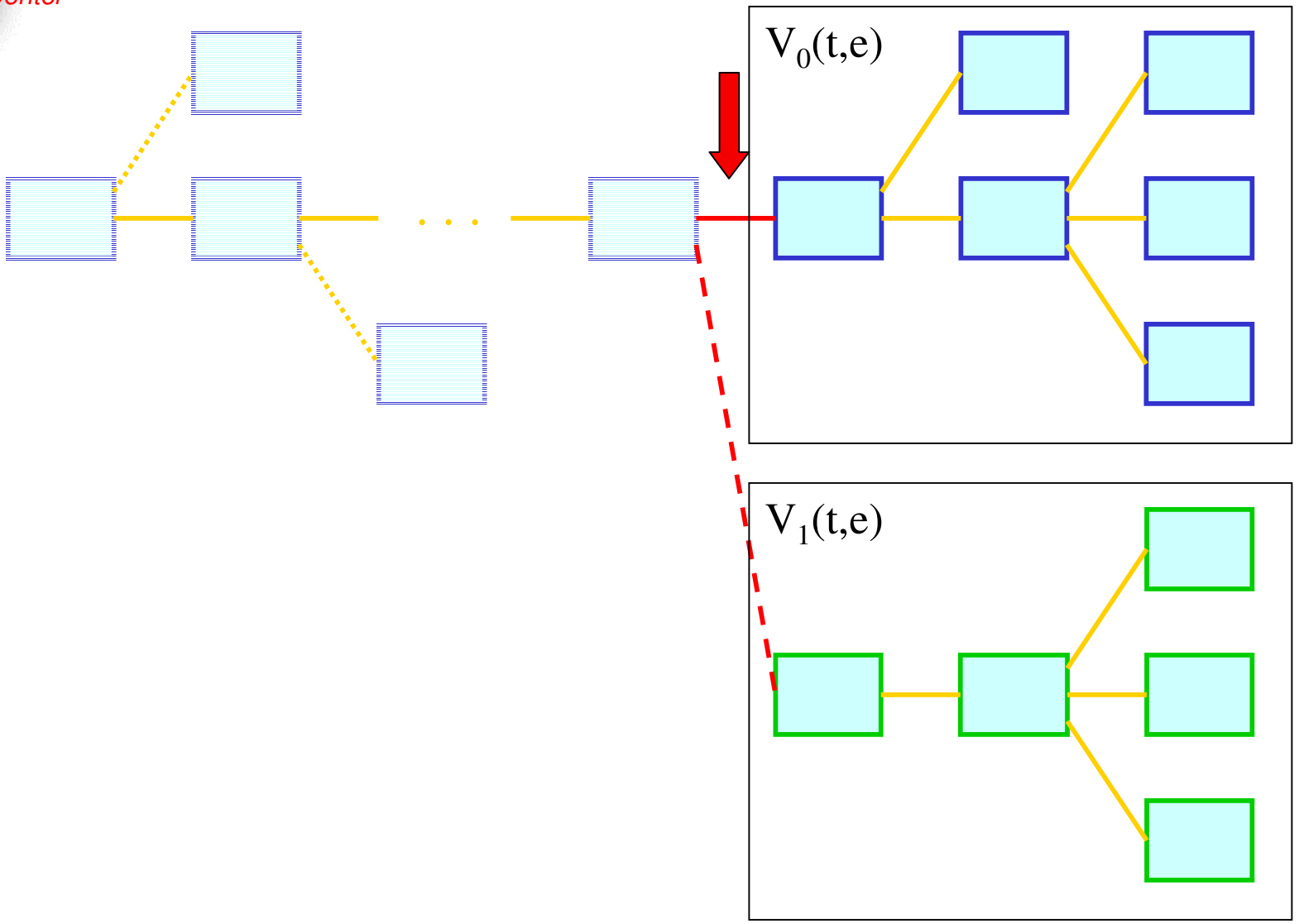
Floating Contingencies



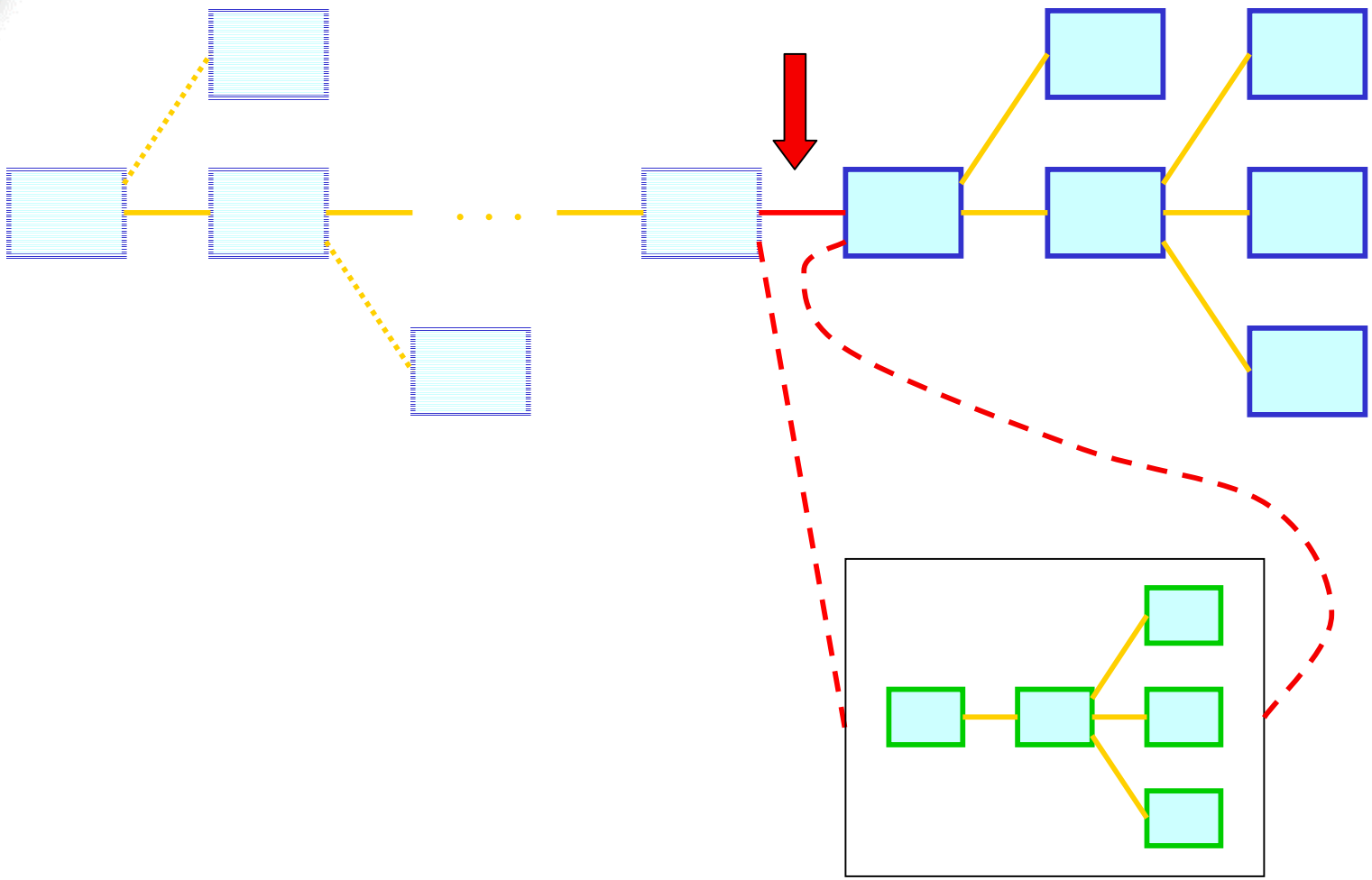
Floating Contingencies



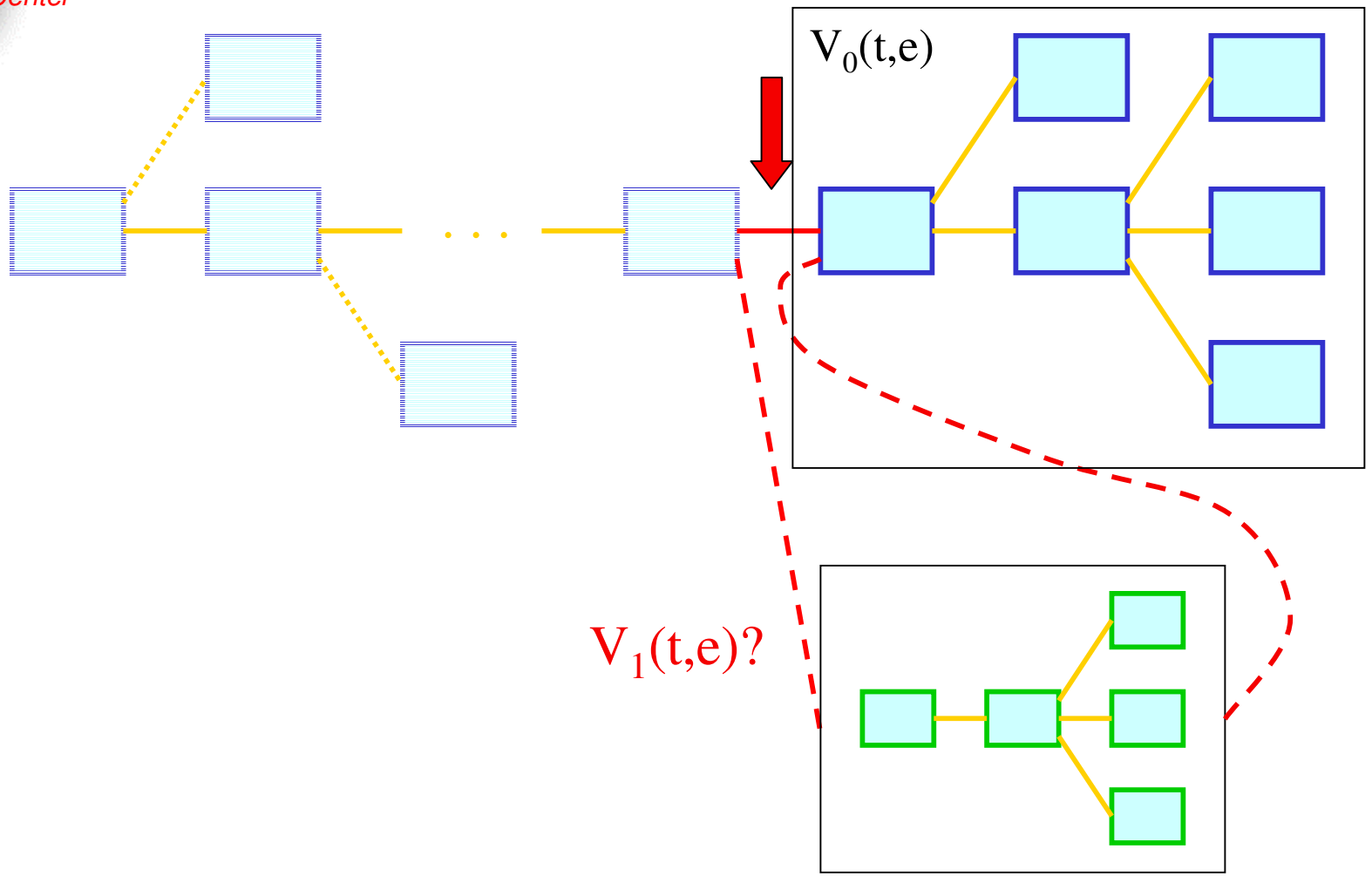
Floating Contingencies – "Replace"



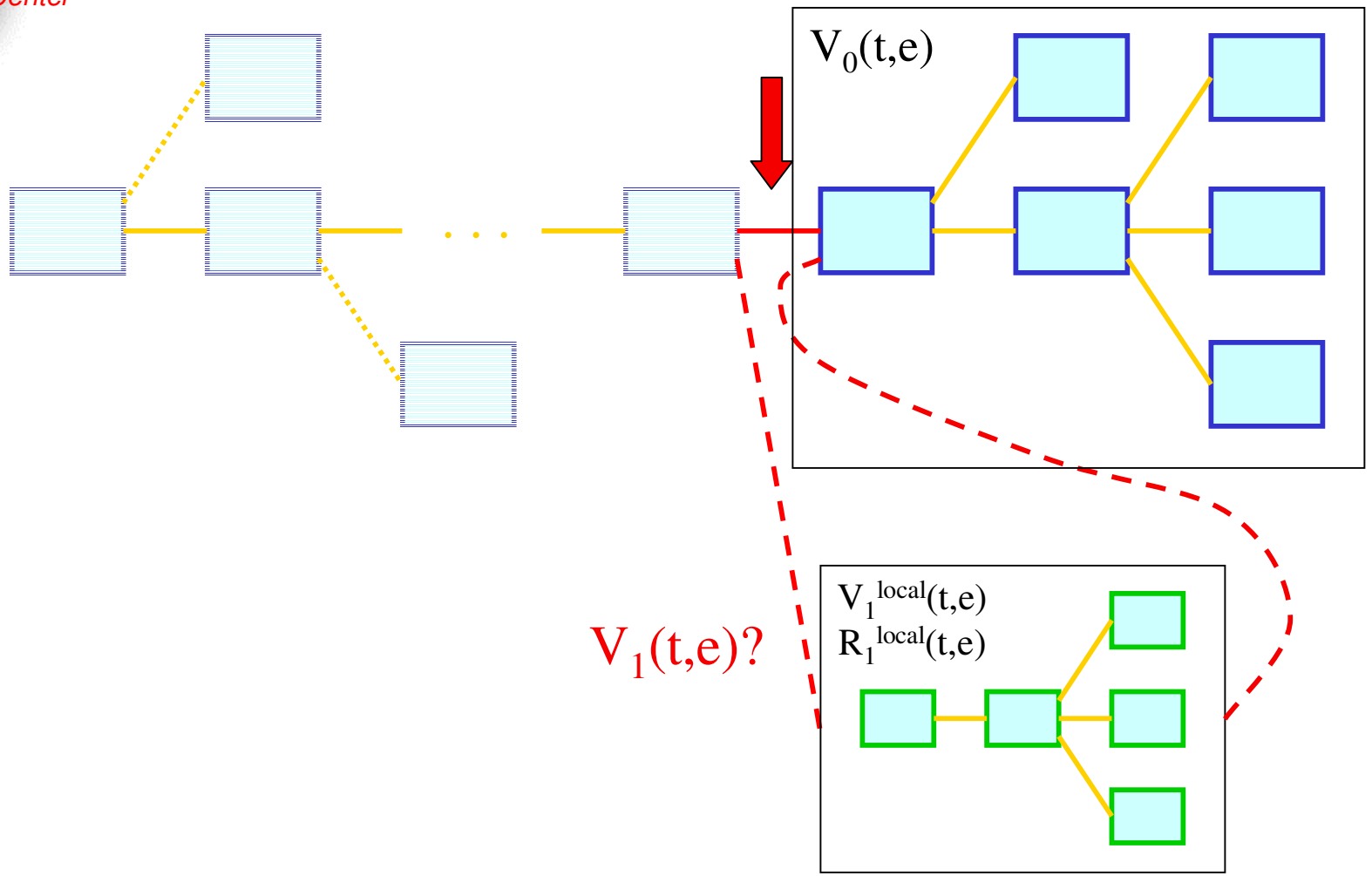
Floating Contingencies – "Insert"



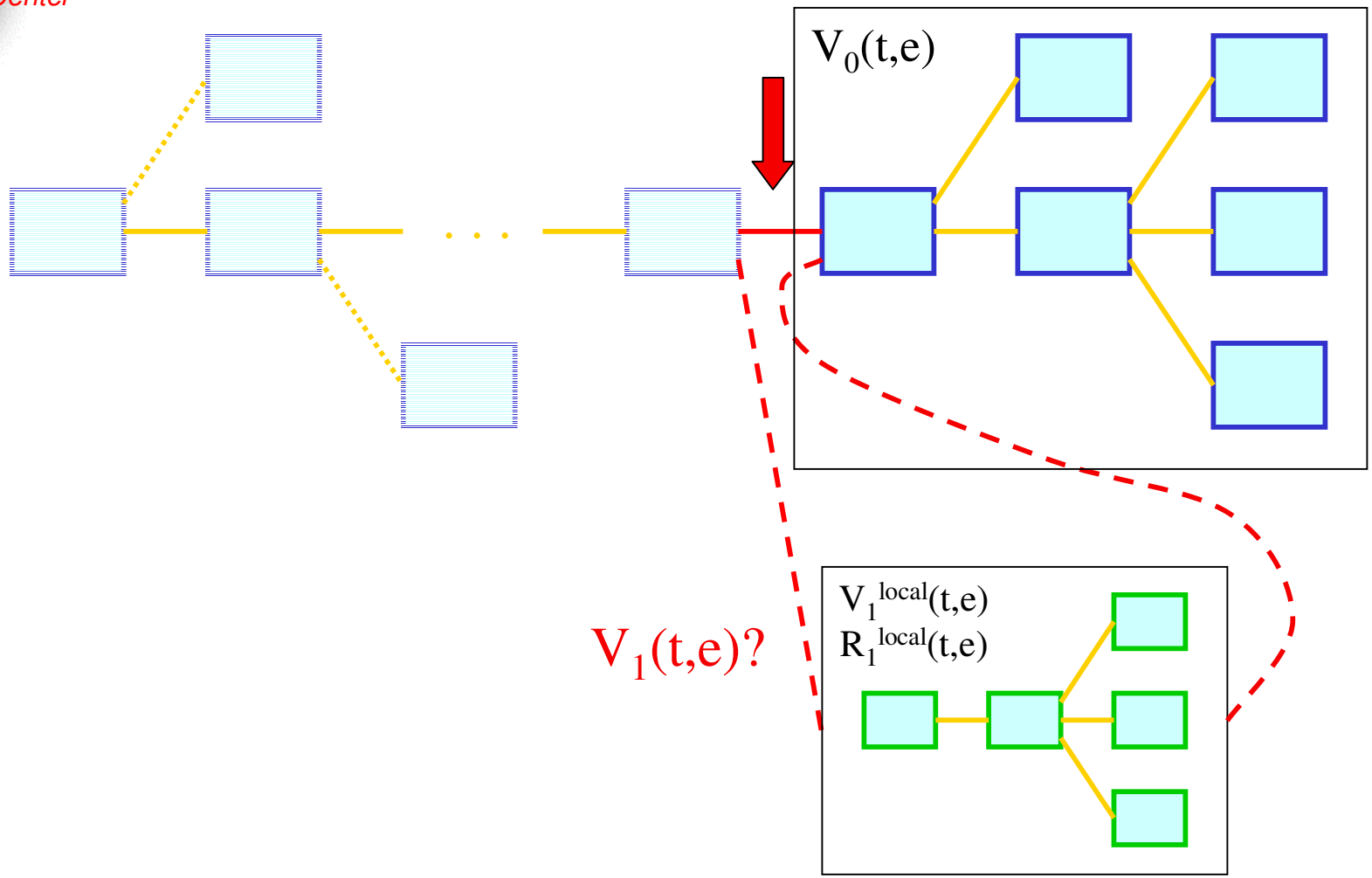
Floating Contingencies – "Insert"



Floating Contingencies – "Insert"

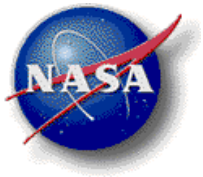


Floating Contingencies – "Insert"

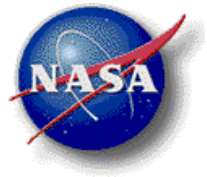


$$V_1(t_0,e_0) = V_1^{local}(t_0,e_0) + \int_t \int_e [P(t,e|t_0,e_0,S_1) \cdot V_0(t,e)]\, dt\, de$$

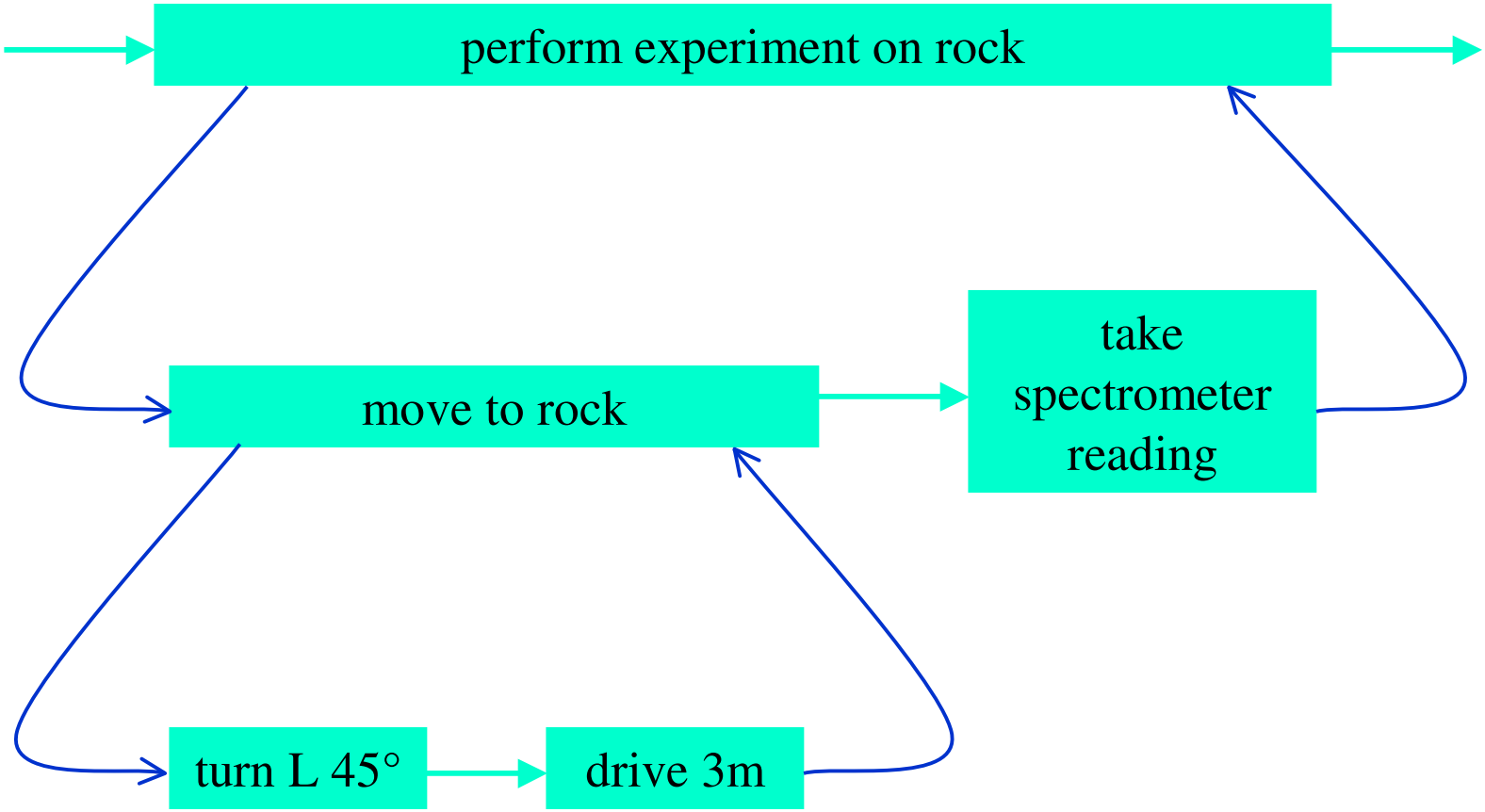
Floating contingencies – issues



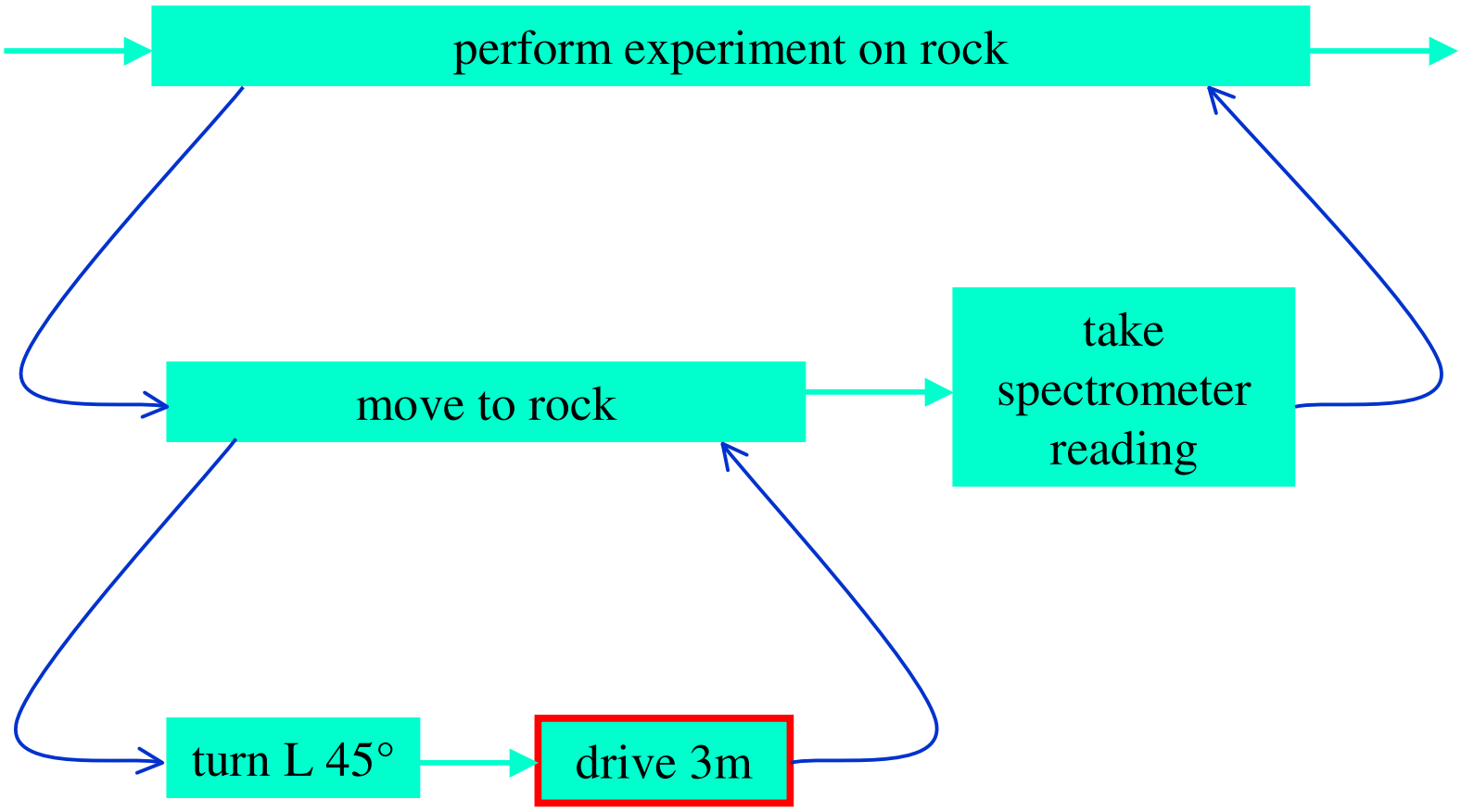
- Interactions between floating contingencies and primary plans
  - Reasoning about preconditions & effects of actions
  - Efficient computation of value function
- Efficient approximations
  - Want to know whether to use floating contingency, not its value
  - Idea: use incremental bounds refinement to handle "obvious" cases quickly
- Recovery plans can vary in locality
  - Use goal structure to direct selection of contingency plans


NASA

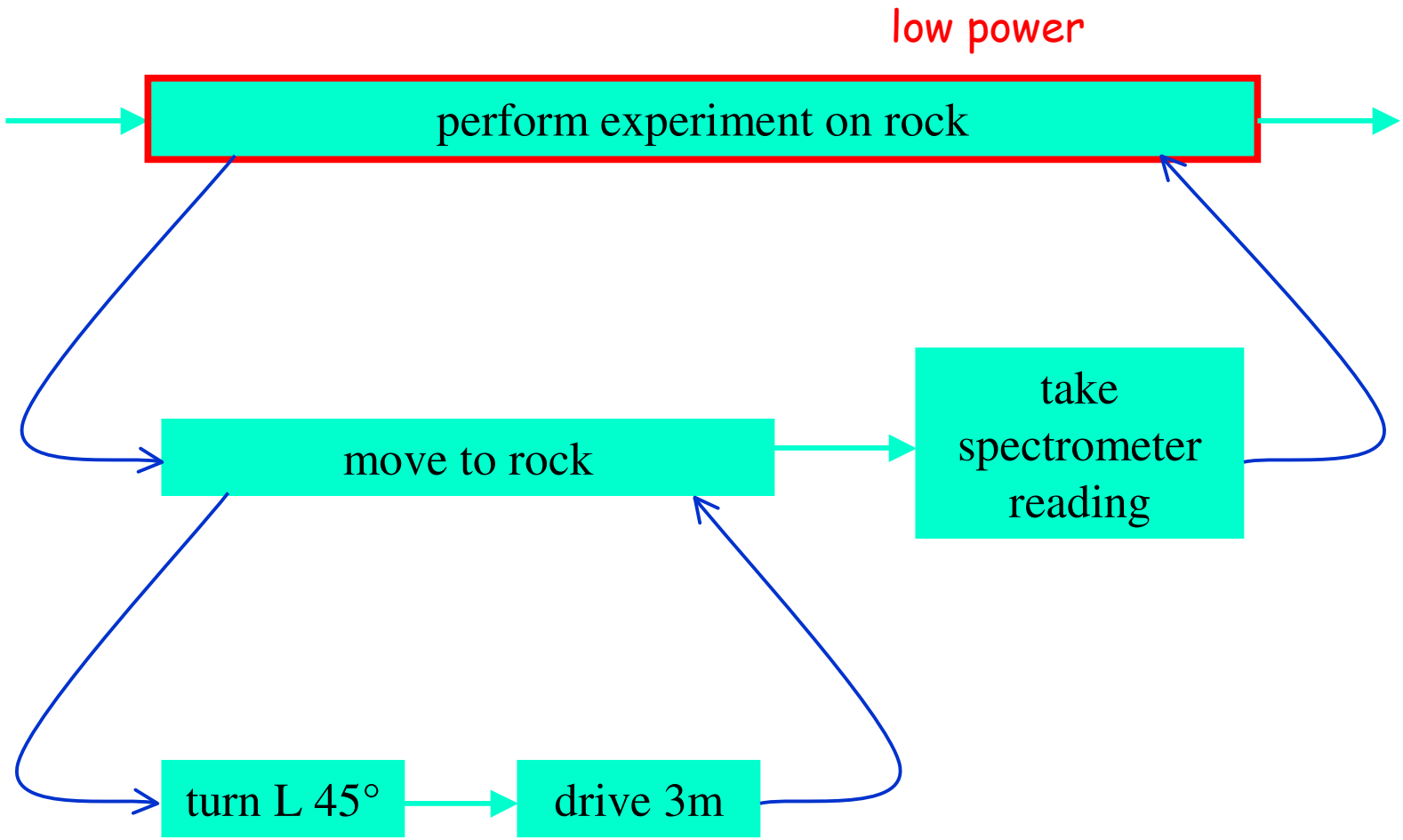
Reacting to contingencies in a hierarchical plan



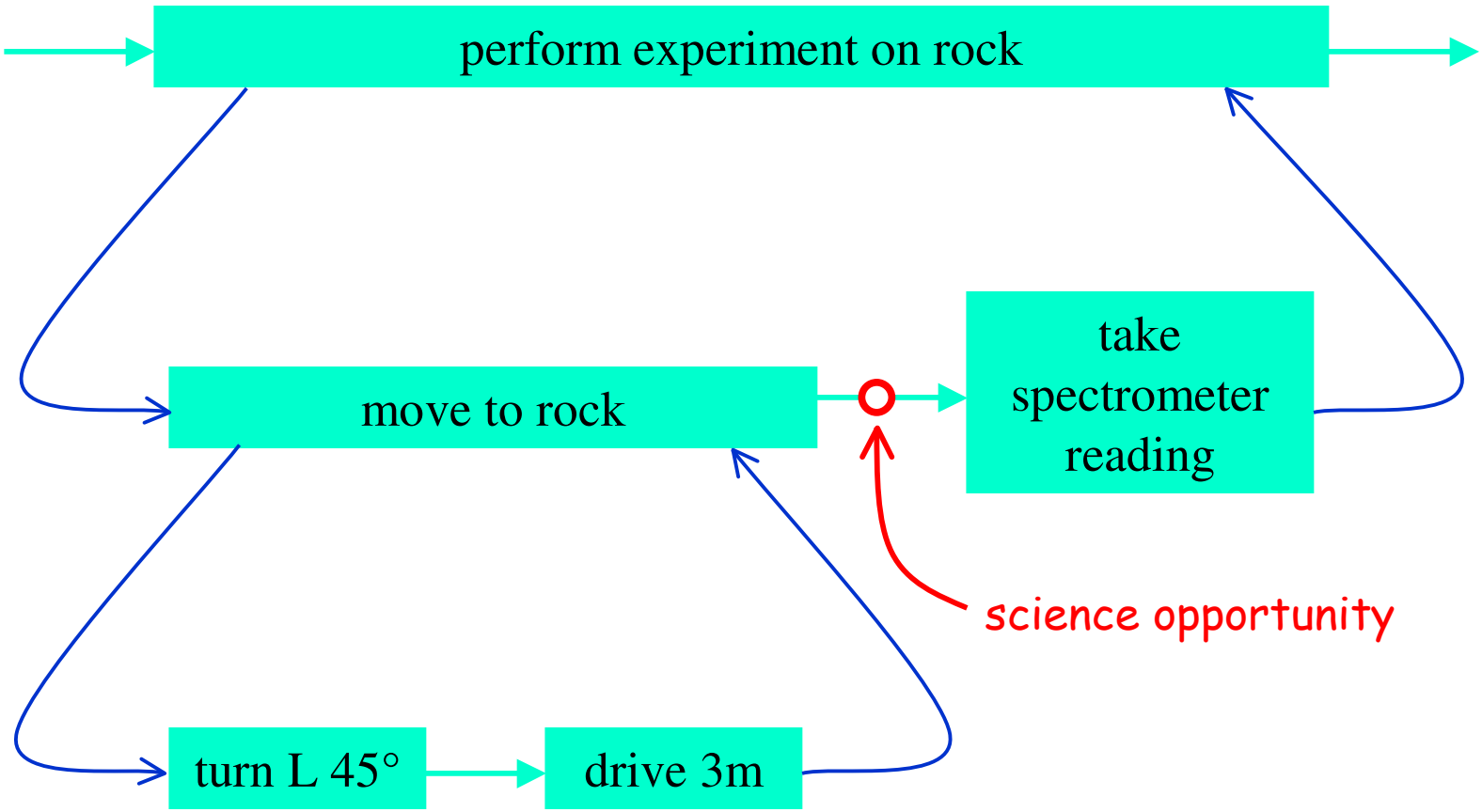
Reacting to contingencies in a hierarchical plan



Reacting to contingencies in a hierarchical plan

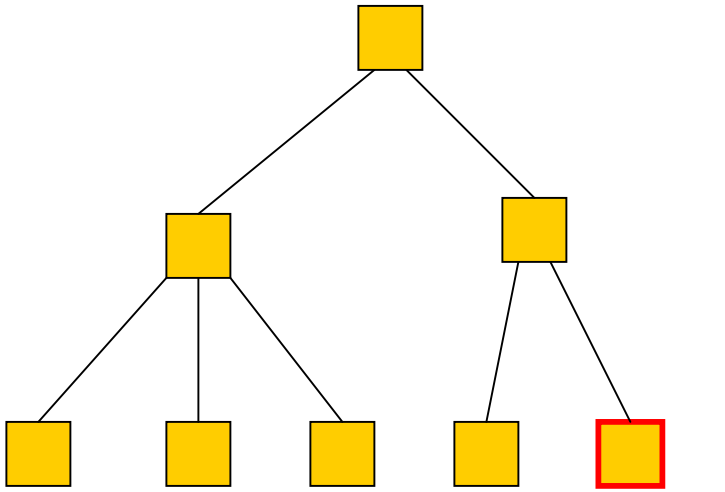


Reacting to contingencies in a hierarchical plan

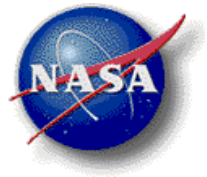


Use of goal structure to direct plan repair

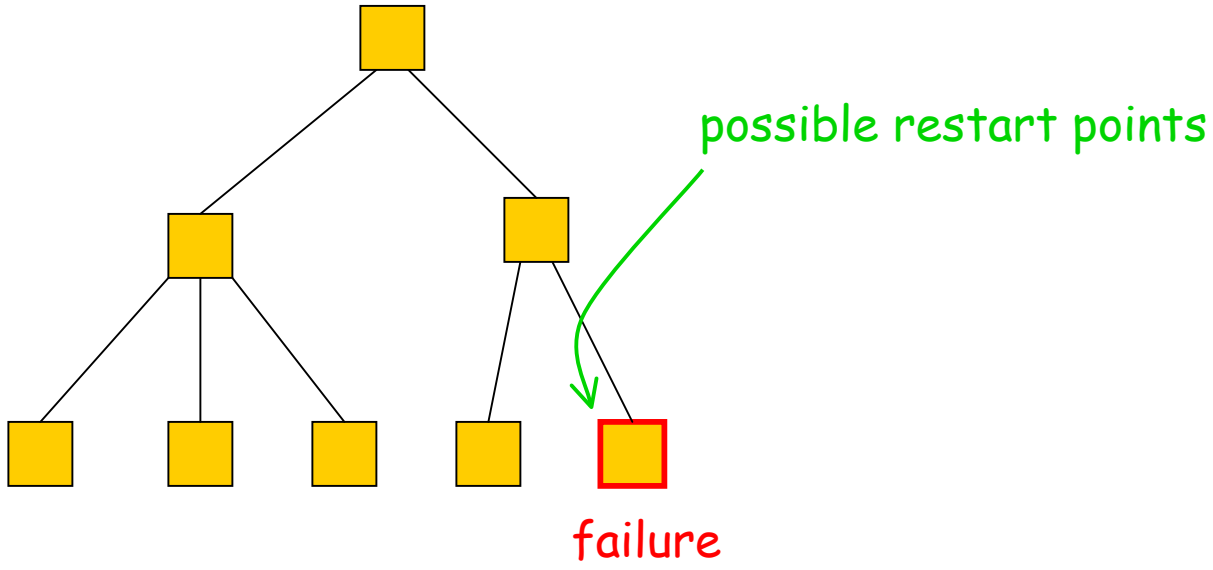
- Goal structure indicates possible places to restart execution after recovery
  - → places to add recovery plans
  - – current point
  - – beginning of any enclosing subgoal



Use of goal structure to direct plan repair

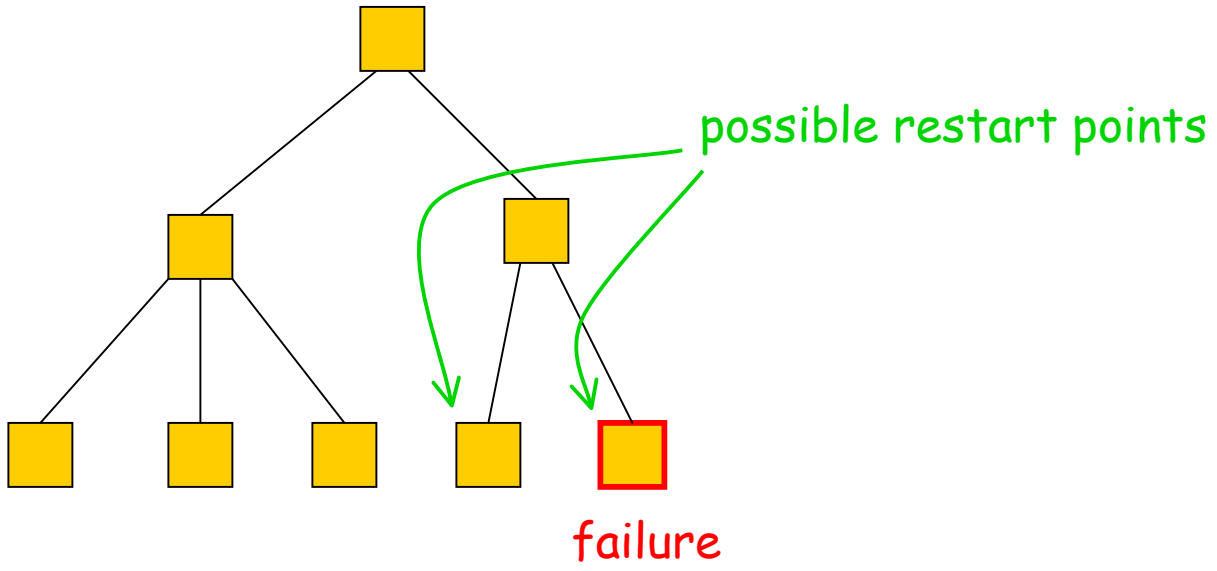


- Goal structure indicates possible places to restart execution after recovery
  - → places to add recovery plans
  - – current point
  - – beginning of any enclosing subgoal

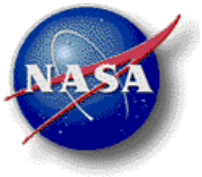


Use of goal structure to direct plan repair

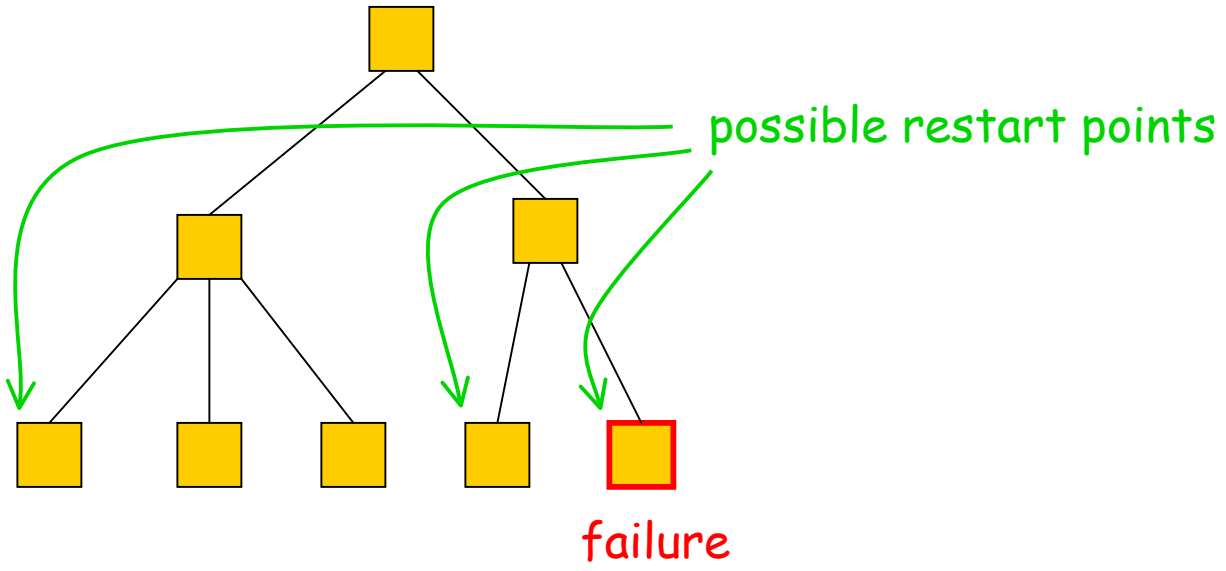
- Goal structure indicates possible places to restart execution after recovery
  - → places to add recovery plans
  - – current point
  - – beginning of any enclosing subgoal

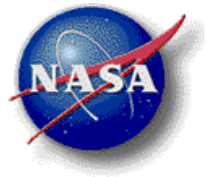


Use of goal structure to direct plan repair

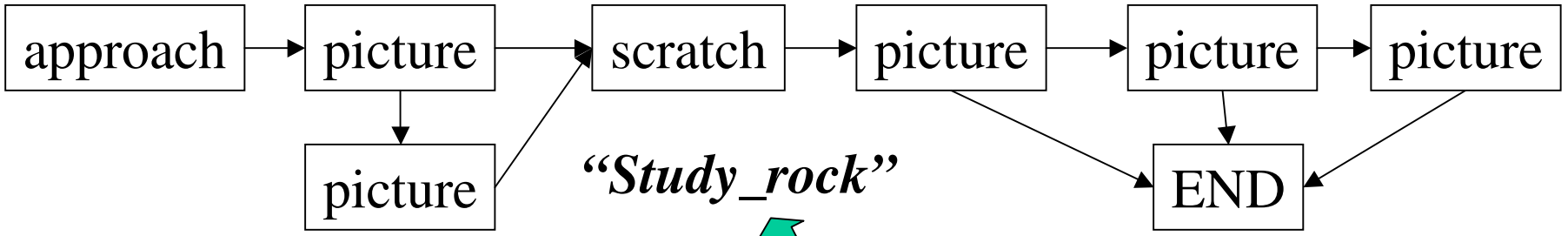


- Goal structure indicates possible places to restart execution after recovery
  - → places to add recovery plans
  - – current point
  - – beginning of any enclosing subgoal

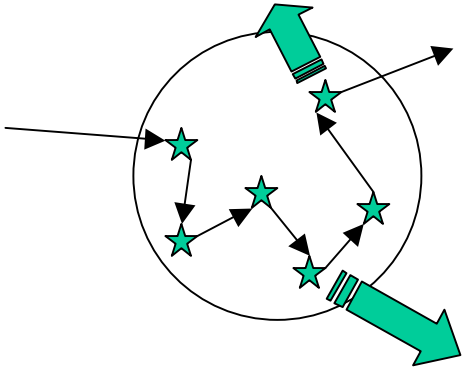



NASA

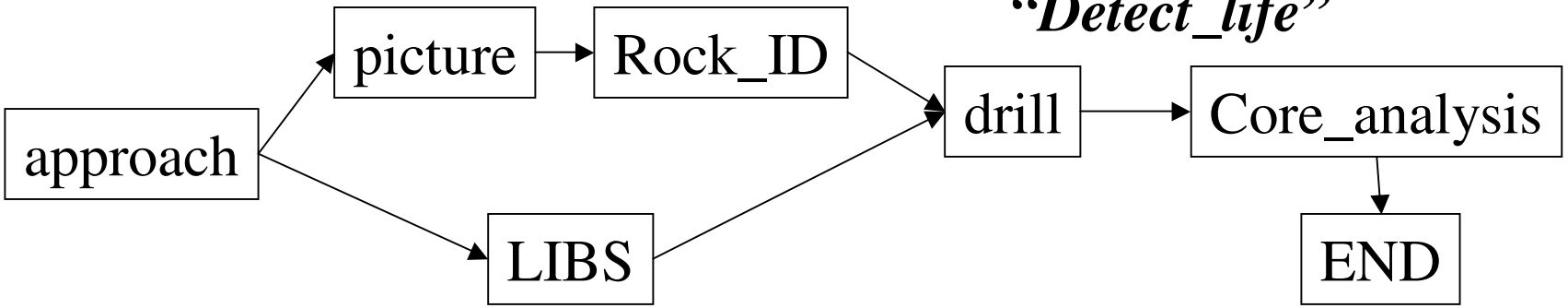
Mission Scenario



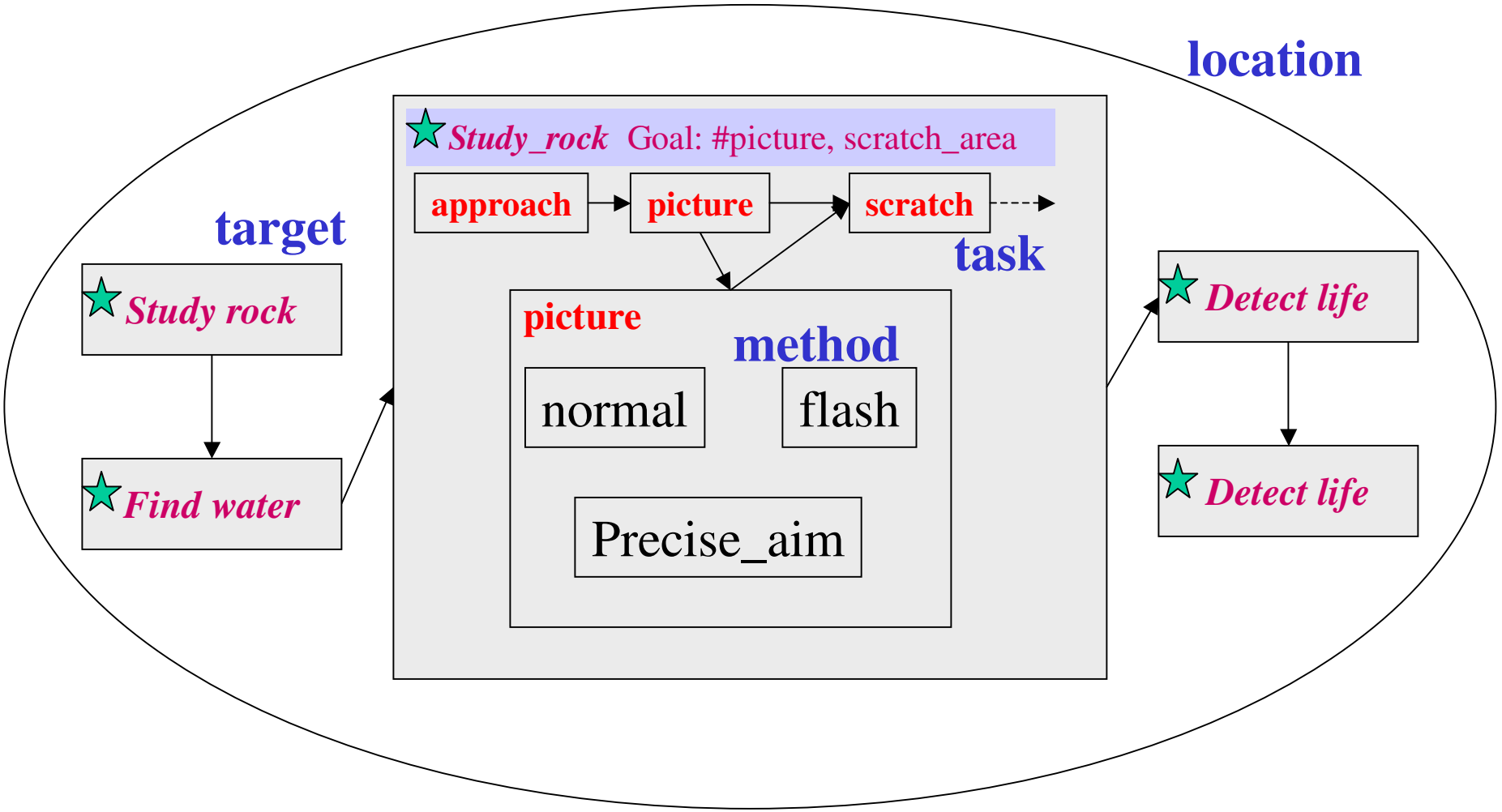
"Study_rock"



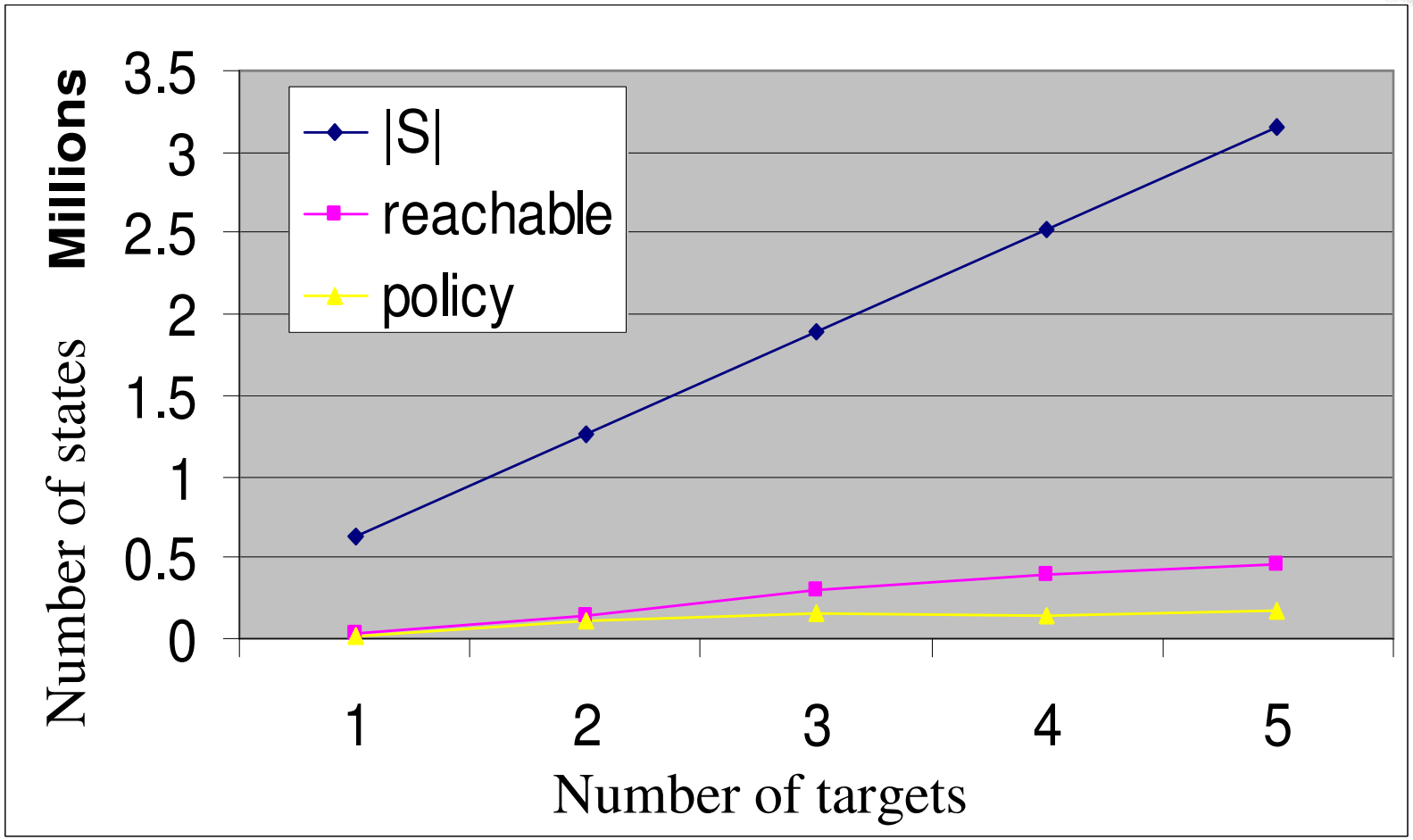
"Detect_life"



Formal Model



Preliminary results



Two resources [0-60], two variables [1-5]
Same task graph at each target

More results

Problem	Size of state space	Reachable states	Size of optimal policy
1	7.65×10^7	5.0×10^5	1.6×10^5
2	7.43×10^7	5.4×10^5	2.1×10^5
3	7.43×10^7	5.4×10^5	2.2×10^5

Two resources [0-60]
Four variables [1-5]
Each problem contains 5 targets